Poster Abstract: BurstMAC — Low Idle Overhead and High Throughput in One MAC Protocol

Matthias Ringwald, Kay Römer
Institute for Pervasive Computing, ETH Zurich, Zurich, Switzerland
Email: {mringwal,roemer}@inf.ethz.ch

***Abstract*— Many sensor network applications feature bursty traffic patterns: after long periods of idle time with almost no network traffic, large amounts of data have to be transmitted reliably and in a timely manner. One example is volcano monitoring [5], where precious high-volume data is generated by rare volcanic eruptions.**

Unfortunately, existing MAC protocols do not sufficiently support such applications with bursty traffic patterns. CSMA protocols such as WiseMAC or SCP-MAP have very low overhead in idle situations, but have high overhead and low throughput under high load due to collisions. In contrast, TDMA protocols such as LMAC can handle high loads without collision, but have low throughput and significant overhead in idle mode [2].

BurstMAC closes this gap by combining low idle overhead with high throughput under load. We present the ideas behind BurstMAC and report initial performance results.

I. Protocol Overview

To achieve the above behavior, BurstMAC combines several techniques which will be outlined in the following. To provide high throughput under load, BurstMAC uses efficient time scheduling and multiple radio channels. However, the control overhead is strictly dependent on the amount of traffic. In idle situations, the overhead comes close to that of a few clear-channel assessments.

A. Collision-free Communication

To avoid collisions, BurstMAC operates in synchronous rounds. The sink node is used as time reference for synchronization. Each node synchronizes to the average time of all nodes which are closer to the time reference than itself. Each round consists of 32 frames. Every frame contains a control section and a data section as depicted in Fig. 1. To maximize throughput and to allow for collision-free communication during the data section, BurstMAC uses 32 interference-free data channels and one control channel. The control section is used for time synchronization, to broadcast other information to all network neighbors, and to assign color ids to nodes. As a result of the latter, each node is assigned a color id $c \in 1..32$ that is unique within two hops. The color id c is used for two purposes. Firstly, the control section of frame c is reserved for the node with color id c, which allows a node to send control messages without collision on the control channel in frame c. Secondly, the node receives data on radio channel c during the data section, coordinating multiple senders to receive data without collisions.

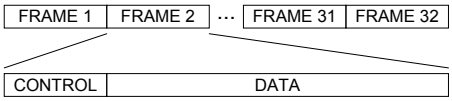

Fig. 1. BurstMAC round consisting of 32 frames which each contain a CONTROL and a DATA section.

B. Coordination-free Transmission Scheduling

During each frame, a node is either in transmit or receive mode, that is, it can either only transmit or only receive data during the whole frame. The choice of mode is controlled by a pseudo-random number sequence which is seeded with the unique 16-bit node id. Knowing the node ids of its neighbors, a node can not only compute its own current mode, but also the current modes of its neighbors. If node A wants to send to neighbor B, then A has to wait for a frame when it is in send mode and B is in receive mode. A uses B's channel for the actual transmission. This approach avoids any extra traffic for coordination among nodes.

C. Cooperative and Single-Bit Transmission

These two physical layer techniques allow for efficient scheduling in the data section of BurstMAC. If multiple senders send a jamming signal at the same time, a receiver can use the Received Signal Strength Indicator (RSSI) to detect that at least one node is sending. This *cooperative transmission* is used to quickly detect if at least one sender wants to send a packet.

To query which nodes want to send, *single-bit transmission* is employed. For this, a coordinating node broadcasts a short synchronization packet to provide a bit-accurate time reference. If node c wants to send a packet, it sends a jamming signal in bit slot c which is detected by the coordinating node. Both techniques are described in more detail in [4].

D. Packet Bursts

To increases throughput and reduce communication overhead, a sender can request the transmission of multiple packets in a row, eliminating lengthy preambles for all but the first packet. Still, each packet has an individual checksum to detect bit errors. The receiver replies a bit vector with a one bit for each packet that has been received correctly.

E. Cross-layer Optimizations

Typical routing protocols such as MintRoute [6] need to perform neighbor discovery and link quality estimation, which

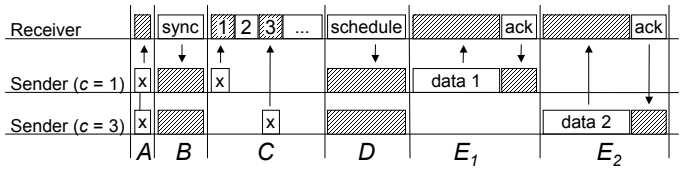

Fig. 2. DATA section with two senders transmitting a packet to the receiver. Cooperative transmission is used in segment A and single bit transmission in segments B and C to identify the senders.

requires each node to broadcast beacon packets at regular intervals. However, due to the existence of the control packets in BurstMAC, we can integrate neighbor discovery and link estimation into BurstMAC without additional overhead.

II. Protocol Details

A. 2-Hop Coloring

Each node has to be assigned a color id which is unique within two hops. For this, all nodes keep track of the frames used by their neighbors for sending control messages. As a node with color id c uses frame c for its control message, each nodes is aware of the colors used by its neighbours and periodically broadcasts a bit vector of these occupied color ids in its control message. A new and uncolored node with id i receives the list of used color ids in the control messages of all of its neighbors. The union of these sets results in the set of color ids used in its 2-hop neighborhood. The new node then randomly picks a color id c from the remaining free colors and transmits its control packet in frame c in the next round. A special flag in the control message requests other nodes to echo the node id contained in the control packet of frame c. If another node with id j in parallel picks the same color c, both node ids i and j will be reported for frame c by different neighbors. In this case, both newly colored nodes pick another free color at random.

B. Efficient Transmission Scheduling

BurstMAC efficiently schedules data transfer on demand using cooperative and single-bit transmission. Fig. 2 shows the data section in more detail, time increases from left to right. One node is in receive mode and two nodes in send mode want to transmit a packet to the receiver. In segment A, both senders employ cooperative transmission and concurrently transmit their send request to the receiver. If there is at least one sender, the receiver exerts the single-bit transmission technique by sending a minimal sync packet in segment B. The sync packet allows a sender to accurately synchronize and send a single jamming "bit" in the slot which corresponds to its color id c in segment C. Based on the list of senders, the receiver than computes and broadcasts the transmission schedule in segment D. In each data segment E_x, a sender transmits a data packet which is immediately acknowledged by the receiver.

C. Network Startup

All nodes concurrently send a short jamming signal, called *Blip*, for 100 us at the very beginning of the control section on an extra Blip channel. By this, a new node joining the network has to scan the control channel at 100% duty-cycle only for a single frame instead of a whole round. On detection of the Blip, the node already has approximate timing information on the start of the control section and will receive at least one control message in the next 32 frames.

III. Implementation and Preliminary Measurements

We implemented BurstMAC on BTnode Rev. 3 nodes [1]. They consist of an ATMEL ATmega128 8-bit microcontroller, 256 KB SRAM, a ChipCon CC1000 radio module and a Zeevo ZV-4002 Bluetooth module. BTnut, an extension of Nut/OS, is used as operating system. The CC1000 module is configured for 34 separate channels (32 for data communication, 1 for control and 1 for Blip broadcasts). We use a frame length of 1 s, which is split into 50 ms for the control section and 950 ms for the data section. The RSSI output is used for clear-channel assessment, cooperative and single-bit transmission. We further make use of the CC1000's ability for precise MAC layer timestamping in the order of 10 us [3].

The energy consumption of BurstMAC in the idle case is dominated by the periodic broadcast of control messages which in turn depends on the number of neighbors. For a well-connected network with seven neighbors, a node runs at 0.8 % duty cycle. This is comparable to other state-of-the-art low-power CSMA MAC protocol such as WiseMAC and SCP-MAC when link estimation and routing messages are included. The duty-cycle for a node which is constantly sending or receiving data, e.g., a neighbor to the sink when all nodes have data to send, can reach 94% while at the same time up to 74% of the raw bandwidth of 19200 baud are utilized for data transport. In other words, 78.7% of the raw bandwidth is used for the actual payload transmission without collisions.

IV. Outlook

We are currently working on a thorough evaluation using 30 BTnodes distributed in our lab. Support for very dense networks is a further topic as 32 channels might not be sufficient for the 2-hop-coloring of very dense networks.

Acknowledgments

The work presented in this poster abstract was partially supported by the Swiss National Science Foundation under grant number 5005-67322 (NCCR-MICS).

References

[1] BTnodes. A distributed environment for prototyping ad hoc networks. www.btnode.ethz.ch.

[2] Koen Langendoen. Medium access control in wireless sensor networks. In H. Wu and Y.Pan, editors, *Medium access control in wireless networks, volume II: practive and standards*. Nova Science Publishers, 2007.

[3] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *SenSys*, 2004.

[4] Matthias Ringwald and Kay Römer. Bitmac: A deterministic, collision-free, and robust mac protocol for sensor networks. In *EWSN*, 2005.

[5] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI*, 2006.

[6] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys*, 2003.